UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Klemen Pernuš

Uporaba odsluženih pametnih telefonov za video nadzor hiše

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Ljubljana, 2016

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Klemen Pernuš

Uporaba odsluženih pametnih telefonov za video nadzor hiše

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

MENTOR: doc. dr. Mojca Ciglarič

Ljubljana, 2016

Tema diplomskega dela

Preglejte področje interneta stvari in preučite, kako lahko vanj umestimo odslužene pametne telefone na platformi Android. Preučite starejše različice Androida in zahteve za razvoj aplikacij, ki bodo tekle na širokem razponu različic tega sistema. Zasnujte sistem za nadzor doma, kjer boste namesto kamer uporabili odslužene pametne telefone in primerno uporabite njihove zmogljivosti, ki so na voljo. Izdelajte načrt sistema, določite funkcionalne specifikacije, sistem implementirajte in preizkusite. Komentirajte izvedbo in možnosti uporabe.

Zahvala

Zahvaljujem se mentorici doc. dr. Mojci Ciglarič za nasvete in pomoč pri izdelavi diplomskega dela.

Kazalo

Seznam uporabljenih kratic

IoT – Internet of Things

NFC – Near-field Communication

RFID – Radio-frequency identification

API – Application Programming Interface

LTS – Long Term Support

JVM – Java Virtual Machine

DVM – Dalvik Virtual Machine

ART – Android Runtime

SDK – Software Development Kit

NDK – Native Development Kit

JNI – Java Native Interface

ADB – Android Debug Bridge

Povzetek

Diplomsko delo se loteva problematike zastarelih in odsluženih pametnih telefonov, ki pa so v večini primerov še vedno polno delujoči. Razvita je bila aplikacija za Android pametne telefone, ki v duhu dobe interneta stvari odslužene telefone spremeni v enote za video in audio nadzor hiše, ki so dosegljive preko lokalnega omrežja. Zaradi želje, da se pokrije čim večje število pametnih telefonov, je bila aplikacija ciljno razvita za starejše verzije Androida. Končna verzija aplikacije omogoča hkratno uporabo več pametnih telefonov, priključenih v lokalno brezžično omrežje. V aplikaciji so pametni telefoni lahko razdeljeni v skupine in znotraj posamezne skupine opravljajo nalogo snemalnika ali nalogo pregledovalnika. Poleg nadzora znotraj lokalnega brezžičnega omrežja omogoča tudi oddaljen nadzor preko interneta.

Ključne besede: odsluženi pametni telefoni, Android, video nadzor

Abstract

The diploma thesis deals with the problem of obsolete smartphones that are in most cases still fully functional. Developed was an application for the Andorid smartphones that turns the smartphones into video surveillance units accessible over the local network. Application was targeted for the older versions of Android operating system with the goal of covering as much of the smartphones as possible. The final version of the application enables the usage of one or more smartphones connected to the local wireless network. Smartphones can be assigned into different groups and inside the group they perform the role of a recorder or of a viewer. The viewers can access the recorders over the local wireless network or over the internet through the proxy server.

Keywords: obsolete smartphones, Android, video surveillance

1. Uvod

Danes trg pametnih telefonov predstavlja več kot polovico vseh mobilnih telefonov. Njihov hiter razvoj je glavni krivec za (pre)hitro zastarelost le-teh. Povprečni uporabnik menja svoj pametni telefon na približno dve leti. V tem času namreč trg (tako strojna kot tudi programska oprema) toliko napreduje, da stari telefon postane prepočasen za uporabo novejših aplikacij ali pa nove aplikacije preprosto niso več združljive z verzijo operacijskega sistema. Kljub vsemu pa je stari (odsluženi) telefon še vedno polno delujoč in posledično uporaben v precej različnih aplikacijah. V diplomski nalogi smo se lotili problematike odsluženih telefonov in jim z razvojem aplikacije poskusili podaljšati uporabno vrednost.

Razvili smo aplikacijo za Android pametne telefone, ki izkorišča dejstvo, da ima vsak pametni telefon vsaj eno kamero, mikrofon in zvočnik ter zmožnost povezovanja v brezžično omrežje. Pametni telefon se tako spremeni v enoto za video in audio nadzor hiše. Aplikacija omogoča hkratno uporabo več pametnih telefonov, priključenih v lokalno brezžično omrežje. Ti so lahko razdeljeni v različne skupine, znotraj vsake skupine pa posamezne naprave opravljajo nalogo snemalnika oziroma pregledovalnika. Naloga snemalnika je zajem videa in zvoka ter zaznavanje dogodkov, kot so: premikanja na videu, prekomerne jakosti zvoka in pritiska na tipko na vhodu za slušalke. Vse to se pošilja pregledovalnikom, katerih naloga je opozoriti uporabnika na zaznano dogajanje in mu omogočiti pregled videa in zvoka.

2. Pregled področja

2.1 Internet stvari (*Internet of Things - IoT*)

Internet stvari je širok pojem, katerega najlažje opišemo kot sistem medsebojno povezanih naprav ter primerno označenih objektov, ki si izmenjujejo podatke. Internet stvari je lahko že preprosta NFC značka, preko katere na mobilnem telefonu nastavimo želene nastavitve, števec korakov, ki se poveže preko Bluetootha na mobilni telefon in preko njega pošilja podatke na strežnik na internetu ali pa je to nadzorni sistem, do katerega lahko dostopamo preko interneta.

Sama ideja povezovanja naprav na internet, z namenom oddaljenega nadzora, je stara že več desetletij. Prva taka naprava je bil avtomat za pijačo na Carnegie Mellon University, ki so ga na internet povezali leta 1982 z namenom preverjanja dostopnosti in temperature pijače [1]. Termin internet stvari je bil prvič uporabljen leta 1999, ko je Kevin Ashton z njim opisal sistem, v katerem bi preko senzorjev na internet lahko povezali vsakodnevne predmete. Z njim je hotel poudariti uporabnost RFID značk za označevanje predmetov z namenom sledenja in beleženja zalog blaga v skladišču [2].

Internet stvari, poleg nadzora in interakcije z napravami, omogoča tudi izgradnjo t.i. pametnih prostorov [8]. Pri pametnih prostorih si naprave, kot so razna tipala, stikala in z značkami opremljeni predmeti, izmenjujejo podatke ter se odzivajo na dogodke. S podatki, zbranimi preko teh naprav, si gradijo bazo podatkov, ki jih hranijo in obdelujejo. S tem je posameznim napravam omogočeno, da postanejo »pametne«, torej da se glede na izkušnje prilagajajo oziroma učijo in s tem izboljšujejo svojo namembnost [6].

2.1.1 Komunikacija in interakcija

Naprave lahko komunicirajo neposredno druga z drugo ali pa pošiljajo podatke in prejemajo sporočila s strežnika na internetu. Možnih je več vrst komunikacijskih povezav [4]:

- povezava naprava-naprava (*device-to-device*):

ta vrsta povezave predstavlja dve ali več naprav, ki se povezujejo neposredno ena na drugo. Naprave lahko med seboj komunicirajo preko lokalnega omrežja ali pa za povezovanje uporabljajo enega od protokolov, kot so Bluetooth, Z-Wave, ZigBee, itd.;

- povezava naprava-internet (*device-to-cloud*):

  pri tej vrsti povezave se naprave povezujejo neposredno na strežnik na internetu ter si preko njega izmenjujejo podatke. Naprave za dostop do interneta izkoriščajo obstoječa lokalna oziroma mobilna omrežja;

- povezava naprava-prehod (*device-to-gateway*):

  ta tip povezave je kombinacija prejšnjih dveh, saj se posamezne naprave povežejo neposredno na posredniško napravo, ki nato njihove podatke zbira in posreduje na strežnik na internetu.

Povezovanje in komuniciranje med samimi napravami lahko poteka brez posredovanja uporabnika. Še vedno pa mora obstajati način interakcije uporabnika s samo napravo za same nastavitve ali morebitni pregled podatkov. Dostop in interakcija z napravami tipično poteka preko spletnega brskalnika oziroma namenske aplikacije. Z brskalnikom oziroma aplikacijo se lahko povežemo neposredno na napravo, v kolikor naprava to omogoča, ali pa se povežemo na strežnik, na katerega naprava pošilja podatke. Prednost uporabe strežnika je v tem, da do podatkov o napravi lahko dostopamo od kjerkoli, kjer imamo dostop do interneta.

2.1.2 Značke (*tags*)

V internet stvari lahko vključimo tudi predmete, ki sami po sebi niso sposobni komunikacije. V ta namen jih opremimo z značkami, ki vsebujejo ime predmeta, njegovo identifikacijsko številko ali pa kar spletni naslov, kjer dobimo informacije o predmetu. S tem nam internet stvari združi fizični svet predmetov z virtualnim svetom podatkov in informacij. Omogoča nam izgradnjo t.i. fizičnega spleta (*Physical web*) [7].

Najpogostejše so kot značke uporabljene črtne kode, med njimi je najpopularnejša QR koda [5], pasivne RFID in NFC značke ter značke, ki temeljijo na Bluetooth Low Energy standardu (BLE) [7]. Uporabnost značk z novejšimi pametnimi telefoni postaja

vse večja, saj so današnji telefoni po večini opremljeni s kamero, NFC čitalcem ter podpirajo Bluetooth komunikacijo, kar jim omogoča, da preberejo podatke iz vseh zgoraj omenjenih vrst značk.

Tako nam pametni telefoni predstavljajo glavno orodje za interakcijo s predmeti fizičnega spleta.

2.2 Android

Android je operacijski sistem, ki je bil prvotno razvit za pametne telefone in tablične računalnike, torej naprave z na dotik občutljivim zaslonom. Danes ga najdemo tudi v pametnih urah, televizorjih, avtomobilih in drugih vgradnih sistemih. Razvilo ga je podjetje Android, Inc, ki pa ga je leta 2005 kupil Google. Prva verzija Androida je bila izdana leta 2008 skupaj s telefonom G1 [13]. Danes je Android najpogostejši operacijski sistem na pametnih telefonih – v letu 2015 je poganjal več kot 80% v tem letu prodanih pametnih telefonov [14].

Razvoj posamezne verzije Androida poteka znotraj podjetja Google, njegova izvorna koda pa je po izidu verzije javno objavljena in dostopna pod Apache 2.0 licenco. Google nove verzije Androida izdaja na 6 do 9 mesecev. Vsaki izdani verziji je dodeljena identifikacijska številka, t.i. API verzija. API verzija pove, katere funkcije so na voljo v posamezni izdaji Androida. API verzija je pomembna za razvijalce Android aplikacij, saj preko nje povedo, s katero verzijo Androida je njihova aplikacija združljiva (1).

2.2.1 Arhitektura

Glavna strojna arhitektura, na kateri teče Android, je ARM (ARMv7 ter ARMv8-A). Podpira pa tudi x86 in MIPS arhitekturo. Od verzije 5.0 Android poleg 32 bitne podpira tudi 64 bitno verzijo vseh podprtih arhitektur.

Programska arhitektura sestoji iz petih delov (Slika 1):

- jedro,
- sistemske knjižnice,
- Androidov izvajalnik kode (Android runtime),
- aplikacijsko ogrodje (application framework),

- aplikacije.

Jedro

Jedro Android sistema je ena izmed dolgoročno podprtih verzij (LTS) Linuxa, kar Android uvršča med eno izmed Linux distribucij. Samo jedro nato Google dopolni s svojimi razširitvami. Kar nekaj teh razširitev, predvsem rešitve s področja varčevanja z energijo, pa je sprejetih nazaj v uradno Linux jedro.

Sistemske knjižnice

Sistemske knjižnice, kot so WebKit, OpenGL, FreeType, SQLite, bionic, itd., so optimizirane in prevedene za posamezno strojno arhitekturo ter s tem omogočajo kar najboljšo izrabo sistemskih virov in hitrost izvajanja.

Androidov izvajalnik kode

Androidov izvajalnik kode skrbi za izvajanje Android aplikacij. Android aplikacije so pisane v Javi, kar pomeni, da jih je potrebno izvajati na navideznem stroju, ki skrbi za preslikavo posameznih ukazov iz Java bytekode v bytekodo, ki se izvede na procesorju. Navidezni stroj Java (JVM) za izvajanje programov uporablja sklad, kar je v primerjavi z uporabo registrov počasneje, saj so potrebni dodatni ukazi za prenos podatkov na in s sklada. Zato so za Android razvili navidezni stroj Dalvik (DVM), ki za izvajanje programov uporablja registre. Zasnovan je bil za učinkovito izvajanje več primerkov navideznega stroja hkrati. DVM se uporablja na Androidu do verzije 5.0. Pri novejših verzijah se uporablja ART, ki ob namestitvi aplikacije le-to prevede v strojno kodo, ki se izvaja neposredno na procesorju.

Aplikacijsko ogrodje

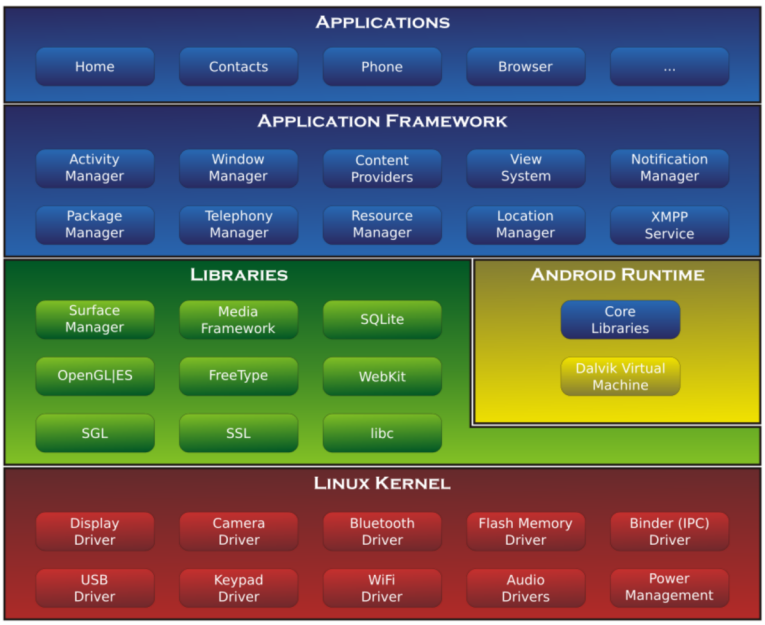
Aplikacijsko ogrodje omogoča dostop do elementov uporabniškega vmesnika, telefona, kontaktov, kamere, itd. Je set knjižnic in razredov, ki se jih uporablja pri razvoju Android aplikacij. Aplikacijsko ogrodje je definirano z API verzijo Androida.

Aplikacije

Aplikacije so vse, kar vidi in s čimer rokuje uporabnik Android naprave.

Verzija androida	Datum izida	API verzija	Ime
Android 6.0	Avgust 2015	23	Marshmallow
Android 5.1	Marec 2015	22	Lollipop
Android 5.0	November 2014	21	Lollipop
Android 4.4W	Junij 2014	20	Kitkat Watch
Android 4.4	Oktober 2013	19	Kitkat
Android 4.3	Julij 2013	18	Jelly Bean
Android 4.2-4.2.2	November 2012	17	Jelly Bean
Android 4.1-4.1.1	Junij 2012	16	Jelly Bean
Android 4.0.3-4.0.4	December 2011	15	Ice Cream Sandwich
Android 4.0-4.0.2	Oktober 2011	14	Ice Cream Sandwich
Android 3.2	Junij 2011	13	Honeycomb
Android 3.1.x	Maj 2011	12	Honeycomb
Android 3.0.x	Februar 2011	11	Honeycomb
Android 2.3.3-2.3.4	Februar 2011	10	Gingerbread
Android 2.3-2.3.2	November 2010	9	Gingerbread
Android 2.2.x	Junij 2010	8	Froyo
Android 2.1.x	Januar 2010	7	Eclair
Android 2.0.1	December 2009	6	Eclair
Android 2.0	November 2009	5	Eclair
Android 1.6	September 2009	4	Donut
Android 1.5	Maj 2009	3	Cupcake
Android 1.1	Februar 2009	2	Base
Android 1.0	Oktober 2008	1	Base

Tabela 1: Verzije Androida in njihove API verzije



Slika 1: Android programska arhitektura

2.2.2 Razvoj aplikacij

Razvoj aplikacij poteka s pomočjo Android paketa za razvoj aplikacij (SDK). Aplikacije so pisane v programskem jeziku Java. Deli aplikacij so lahko pisani v programskem jeziku C/C++, s katerim se razvije dinamična knjižnica, ki se prevede za posamezno strojno arhitekturo. Do funkcij te dinamične knjižnice se iz Jave dostopa s pomočjo JNI vmesnika. Za razvoj v programskem jeziku C/C++ poleg SDK-ja potrebujemo še paket za razvoj aplikacij za strojno arhitekturo (NDK).

Pri razvoju aplikacije določimo tri atribute:

1. *minSdkVersion,* ki določa minimalno API verzijo, s katero je aplikacija združljiva. Ta atribut mora biti nastavljen, saj glede na njegovo vrednost Android dovoli ali zavrne namestitev aplikacije. V primeru, da uporabnik želi namestiti aplikacijo na napravo, ki ima nižjo API verzijo kot aplikacija, bo Android to namestitev zavrnil;
2. *targetSdkVersion,* ki določa API verzijo, za katero je aplikacija razvita. Atribut obvesti sistem, da je bila aplikacija testirana z določeno verzijo Androida. Če atribut ni nastavljen, privzame vrednost *minSdkVersion* atributa;
3. *compileSdkVersion,* ki določa API verzijo, s katero naj se aplikacija prevede.

Android SDK verzije so vnaprej združljive, kar pomeni, da se aplikacijo, ki ima nastavljeno vrednost minimalno podprte verzije (*minSdkVersion*) na 14, lahko namesti na vse naprave, ki imajo API verzijo večjo ali enako 14. Izbira minimalno podprte verzije tako pomeni izbiro med številom podprtih naprav (nižja številka – več podprtih naprav) in dostopom do uporabnejših lastnosti in zmožnosti, ki jih ponujajo novejše API verzije.

Za določene lastnosti, predvsem za elemente uporabniškega vmesnika, Android ponuja združljivostne knjižnice (*Android Support Library*). Te knjižnice vsebujejo klice funkcij, implementiranih v novejših API verzijah, ki so implementirani s pomočjo funkcij, na voljo v starejših API verzijah. To omogoča uporabo elementov uporabniškega vmesnika, ki so bili predstavljeni v verziji Lolipop (API 21) tudi na starejših verzijah Andoida (če je *minSdkVersion* nastavljen na manj kot 21).

2.3 Video-nadzorni sistemi

Video-nadzorni sistemi se uporabljajo za nadzor posameznih prostorov oziroma celotnih objektov. Danes na trgu obstaja zelo veliko sistemov, pri katerih je najpreprostejši sistem že ena IP kamera, ki se poveže v lokalno brezžično omrežje in se jo krmili ter nadzoruje preko internetnega brskalnika iz računalnika v istem lokalnem omrežju. Primer take kamere je Foscam FI8909W kamera (http://www.foscam.com/product/22.html). Profesionalni nadzorni sistemi so tipično sestavljeni iz več kamer, katere so nameščene v zunanjih in notranjih prostorih ter povezane na večkanalni digitalni video snemalnik. Video snemalnik je povezan tako v lokalno omrežje kot neposredno na internet. Omogoča nadzor tako na monitorjih, priključenih neposredno na snemalnik, kot na računalnikih znotraj lokalnega omrežja in oddaljen nadzor preko interneta. Video snemalnik poleg shranjevanja video signala, le-tega analizira za morebitno gibanje in ob zaznanem gibanju obvesti nadzornika. Primer takega sistema je DH-DVR1604LE-AS (http://www.dahuasecurity.in/products/dvr0804-1604le-as--63.html).

2.3.1 Video kamere

Video kamera zajema video signal področja, ki ga nadzoruje. Kamere se delijo na analogne in IP kamere. Analogne kamere zajeti video signal pošiljajo preko analognega kompozitnega izhoda in potrebujejo snemalnik oziroma monitor s primernim vhodom. IP kamere pa se povežejo v lokalno omrežje in preko njega prenašajo digitaliziran video signal. Za pregled videa je dovolj internetni brskalnik, s katerim se povežemo na posamezno kamero. IP kamere se v lokalno omrežje povezujejo preko žične, nekatere pa tudi preko brezžične povezave.

Pomembne lastnosti video kamer so:

- vidni kot:

  določa, kolikšen del prostora pokrije posamezna kamera. Če ima kamera majhen vidni kot, jih je za pokritje področja lahko potrebnih več kot ena;

- ločljivost zajete slike:

višja kot je ločljivost, bolj natančna je zajeta slika, kar pomeni, da so objekti na sliki lažje razločljivi. To naprednejšim sistemom omogoča boljšo prepoznavo obrazov, prepoznavo registrskih tablic, ipd.;

- nočni vid:

  omogoča nadzor področja tudi pri malo oziroma nič svetlobe. Nočni vid je tipično implementiran z LED IR diodami, ki osvetljujejo okolico kamere. Pri tem je pomembno, kolikšno razdaljo so IR diode sposobne osvetliti;

- zajem zvoka:

  omogoča, da se poleg slike zajema še zvok;

- namen uporabe:

  zunanja ali notranja uporaba. Zunanje kamere morajo biti dodatno zaščitene pred vremenskimi vplivi.

2.3.2 Video snemalnik

Video snemalnik nam omogoča, da si video signal iz kamer shranimo za kasnejši pregled. Če se v video-nadzornem sistemu uporabljajo analogne kamere, je potrebno imeti strojno enoto (digitalni video snemalnik), ki podpira procesiranje analognega video signala. Video snemalniki so tipično večkanalni (4,8,16), kar omogoča priklop več kamer hkrati. V primeru uporabe IP kamer lahko za procesiranje uporabimo programsko opremo, ki teče na osebnem računalniku. Primer programske opreme, ki omogoča priklop na več IP kamer in procesiranje zajetega videa, je Zoneminder (https://zoneminder.com/) oziroma preprostejši Motion (https://github.com/Motion-Project/motion).

Video snemalniki omogočajo več načinov shranjevanja videa:

- stalno shranjevanje:

  celotni video signal se shrani. Ta možnost je prostorsko najbolj potratna, poleg tega je za sam pregled dogajanja v nadzorovanem področju potrebno pregledati precejšno količino shranjenega video zapisa;

- shranjevanje po urniku:

video signal se shranjuje samo ob vnaprej določenem časovnem obdobju;

- shranjevanje ob premikanju:

  video snemalni sistem analizira vhodni video signal in ga ob zaznanem gibanju začne shranjevati. Ta možnost je prostorsko najoptimalnejša, saj imamo shranjen le tisti del video signala, na katerem je neko dogajanje v nadzorovanem območju. Poleg tega nam že samo število posnetkov da informacijo o količini dogajanja v nadzorovanem območju.

2.3.3 Nadzor

Pregled dogajanja in nadzor se izvajata preko monitorja, priklopljenega neposredno na digitalni video snemalnik, oziroma preko programa na osebnem računalniku, če je video snemalnik ali video kamera priklopljena v lokalno omrežje. Nadzor je možen tudi preko oddaljenega dostopa, če sistem to dovoljuje. Nekateri sistemi vsebujejo tudi aplikacije za pametne telefone, preko katerih lahko dostopamo do trenutnega videa oziroma pregledujemo shranjene video posnetke. Sistemi poleg samega videa omogočajo tudi obveščanje o izrednih dogodkih. Ti so lahko zaznano premikanje oziroma presežena jakost zvoka, izguba video signala, pojavitev alarma (če sistem podpira alarmne vhode), …

3. Uporabljene tehnologije

V svetu pametnih telefonov prevladujejo trije operacijski sistemi: Android, iOS in Windows Phone. Vsi operacijski sistemi imajo prosto dostopna orodja za razvoj aplikacij, dostopno dokumentacijo in dobro podporo razvijalcem. Pri vsakem poteka razvoj v drugačnem programskem jeziku, ki so med seboj nezdružljivi – tako s stališča aplikacij kot tudi s stališča programskih knjižnic.

To je bila naša prva aplikacija za mobilne telefone in zaradi tega nismo imeli nobene osebne preference glede razvojnega okolja, smo pa imeli dostop do kar nekaj odsluženih pametnih telefonov z Android operacijskim sistemom. Zaradi tega in zaradi dejstva, da je Android trenutno najbolj razširjen operacijski sistem v svetu pametnih telefonov, smo se odločili za Android platformo.

3.1 Razvoj Android aplikacij

Razvoj Android aplikacij poteka v programskem jeziku Java, zato moramo imeti nameščen paket za razvoj Java programov (Java Developement Kit). Ostala orodja, ki so potrebna za izgradnjo aplikacije, dobimo z namestitvijo paketa za razvoj Android aplikacij (Android SDK) oziroma z namestitvijo Android Studia.

Android Studio je prirejena verzija programa InteliJ IDEA, optimizirana za delo z Android telefoni, ki nam:

- omogoča enostavno dodajanje izvornih datotek in drugih virov v aplikacijo,
- nudi urejevalnik izvorne kode, ki omogoča napredno navigiranje po kodi in s sprotnim prevajanjem opozarja na sintaktične napake že med pisanjem kode,
- omogoča izgradnjo aplikacije po delih, kjer se prevedejo samo tiste datoteke, ki so bile spremenjene od prejšnje izgradnje,
- nudi vgrajen sistem za verzioniranje,
- omogoča preprosto uporabo razhroščevalnika,
- omogoča preprosto izgradnjo in zagon aplikacije.

Pri razvoju aplikacij nam Android Studio nudi avtomatizirano izgradnjo aplikacijskega paketa ter avtomatizirano uporabo orodja ADB (*Android Debug Bridge*). ADB omogoča namestitev in zagon aplikacije na Android napravi ter spremljanje zapisov v dnevniško datoteko oziroma zagon razhroščevalnika. Aplikacijo lahko poganjamo na simulatorju oziroma na dejanski Android napravi.

Za uporabo simulatorja s pomočjo upravljalnika navideznih naprav (*AVD Manager*) zgradimo želeno navidezno napravo. Navidezna naprava je lahko katerega koli tipa (telefon, tablica, TV, …) in na njej lahko teče poljubna verzija Androida. Prednost simulatorja je v tem, da lahko testiramo verzije Andorida, za katere nimamo dostopa do dejanskih naprav.

Za uporabo dejanske naprave moramo na napravi omogočiti način razvijalca. S tem dovolimo, da se na napravo lahko povežemo z orodjem ADB. To storimo v sistemskih nastavitvah naprave. Napravo nato priključimo preko USB priključka na računalnik in namestimo ustrezne gonilnike.

Pri testiranju aplikacij na Android napravah nam Android Studio omogoča, da aplikacijo namestimo in poženemo na več kot eni napravi hkrati. Ta lastnost je uporabna pri razvoju strežnik/odjemalec aplikacij, kjer želimo hkrati spremljati dogajanje tako na strežniku kot tudi na odjemalcu.

3.2 Zgradba Android aplikacije

Android aplikacijo sestavljajo štirje osnovni gradniki:

- aktivnosti (*Activity*),
- storitve (*Services*),
- prejemniki sporočil (*Broadcast Receivers*),
- ponudniki vsebine (*Content Providers*).

Vsi osnovni gradniki, ki sestavljajo aplikacijo, morajo biti opisani v manifest datoteki *AndroidManifest.xml*. V manifest datoteki so opisane lastnosti posameznih gradnikov, povezave med njimi in zahtevana dovoljenja. Android aplikacija namreč privzeto nima dostopa do vseh virov in naprav v telefonu. Za vsak želen vir, kot je na primer uporaba

kamere, uporaba interneta, dostop do dokumentov, itd., se mora zahtevati dovoljenje za uporabo. Uporabo želenih virov omogoči uporabnik ob namestitvi aplikacije na telefon.

3.2.1 Aktivnosti (*Activity*)

Aktivnost predstavlja posamezno okno uporabniškega vmesnika in skrbi za uporabnikovo interakcijo z aplikacijo. Vsaka aktivnost skrbi za svoj vidik aplikacije. Na primer aplikacija za e-pošto ima eno aktivnost, ki prikaže spisek prejetih sporočil, drugo aktivnost, ki pokaže posamezno sporočilo v celoti, in tretjo aktivnost, kjer se napiše novo sporočilo. Če ima aplikacija več kot eno aktivnost, mora ena izmed njih biti označena kot glavna aktivnost, ki se prikaže ob zagonu aplikacije.

3.2.2 Storitve (*Services*)

Storitve so procesi, ki tečejo v ozadju in so aktivni, tudi ko aplikacija ni trenutna aplikacija na zaslonu. Uporabljajo se za dalj časa trajajoče operacije. Na primer, storitev lahko v ozadju predvaja glasbo oziroma lahko pretaka vsebino preko omrežja, medtem ko uporabnik uporablja drugo aplikacijo.

3.2.3 Prejemniki sporočil (*Broadcast Receivers*)

Prejemniki sporočil omogočajo komunikacijo med aplikacijo in operacijskim sistemom ter drugimi aplikacijami. Preko njih operacijski sistem sporoča dogodke, kot so: priklop slušalk, priklop napajanja, pritisk tipke za glasnost, itd.

3.2.4 Ponudniki vsebine (*Content Providers*)

Ponudniki vsebine omogočajo dostop do podatkov med aplikacijami oziroma dostop do baze podatkov.

3.3 Življenjski cikel gradnika

Vsak od osnovnih gradnikov ima svoj življenjski cikel, s katerim upravlja operacijski sistem. Upravljanje poteka preko povratnih klicev, s katerimi operacijski sistem gradniku sporoča, kaj v kakšnem stanju je. Najpomembnejši osnovni gradnik je aktivnost, saj predstavlja povezavo med uporabnikom in aplikacijo. Temu primerno ima tudi najbolj obsežen življenjski cikel sestavljen iz sledečih povratnih klicev (Slika 2):

- onCreate():

je prvi povratni klic in se kliče, ko je aktivnost prvič ustvarjena,

- onStart():

  se kliče, ko aktivnost postane vidna za uporabnika,

- onResume():

  se kliče, ko aktivnost postane aktivna (uporabnik lahko začne z uporabo),

- onPause():

  se kliče, ko postane aktivna druga aktivnost,

- onStop():

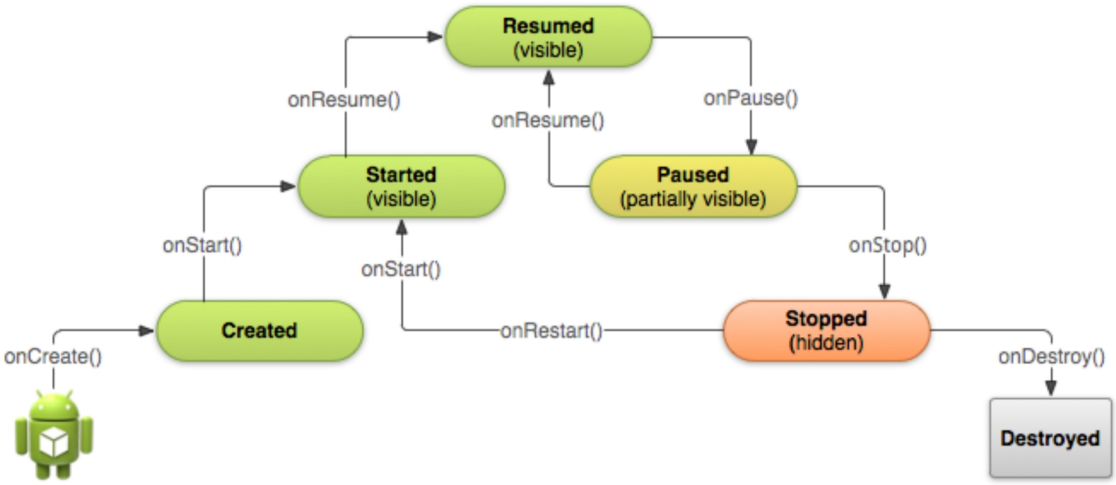
  se kliče, ko aktivnost ni več vidna na ekranu,

- onRestart():

  se kliče, ko se ustavljena aktivnost ponovno požene,

- onDestroy():

  se kliče, preden operacijski sistem uniči aktivnost.



Slika 2: Življenjski cikel aktivnosti [10]

4. Načrt sistema

4.1 Komponente

Naš cilj je bil razviti aplikacijo za pametne telefone, s katero bi telefone lahko uporabili za video nadzor hiše. Aplikacija naj bi omogočala, da telefon uporabimo kot nadzorno kamero ter preko drugega telefona izvajamo nadzor. Povezovanje med telefoni naj bi bilo omogočeno tako lokalno, znotraj istega omrežja, kot oddaljeno, preko interneta.

Razvit sistem je sestavljen iz aplikacije in posredniškega strežnika. Aplikacija sama je sestavljena iz snemalnika in pregledovalnika.

Snemalnik skrbi za zajem videa in zvoka ter spremlja stanje tipke na vhodu za slušalke. Vsak snemalnik je dodeljen določeni skupini, kateri pošilja zajeti video in zvok ter jo obvešča o zaznanem premikanju, prekomerni jakosti zvoka oziroma pritisnjeni tipki.

Pregledovalnik se poveže na snemalnike iz svoje skupine. Vsebuje seznam povezanih snemalnikov, na katerem so prikazana opozorila o zaznanih dogodkih na posameznem snemalniku. Z izbiro posameznega snemalnika se vklopi monitor, ki predvaja video in zvok, zajet na izbranem snemalniku. Iz pregledovalnika lahko omogočimo dvosmerni prenos zvoka med pregledovalnikom in izbranim snemalnikom.

Povezovanje pregledovalnikov in snemalnikov poteka neposredno preko lokalnega omrežja. Vsak pregledovalnik se poveže z vsemi snemalniki, ki pripadajo njegovi skupini. Poleg lokalnega povezovanja pa se vsi snemalniki in pregledovalniki lahko povežejo še na posredniški strežnik, ki pa se nahaja izven lokalnega omrežja.

Posredniški strežnik vsebuje seznam skupin ter sezname snemalnikov in pregledovalnikov, povezanih v posamezno skupino. Njegova naloga je, da podatke, prejete iz posameznih snemalnikov, posreduje na vse pregledovalnike znotraj skupine. Preko posredniškega strežnika je uporabniku omogočen oddaljeni nadzor.

4.2 Tehnična dokumentacija

V aplikaciji je snemalnik predstavljen z razredom BMaster, pregledovalnik pa z razredom BClient.

4.2.1 Razred BMaster

Razred BMaster (Slika 3) vsebuje objekte za upravljanje s kamero (*BCamera*), mikrofonom (*BMic*) in zvočnikom (*BSpeaker*), objekt, ki skrbi za mrežno komunikacijo (*BmCommunicator*), ter objekt, ki skrbi za prikaz slike na aktivnosti (*BImageView*).

Iz razreda BCamera prejema zajeto sliko, ki jo analizira za morebitno gibanje in ob zaznanem gibanju na pregledovalnike pošlje kontrolno sporočilo »`Event=Motion`«. Sliko nato prikaže na aktivnosti, jo zakodira v JPG format ter pošlje na pregledovalnike.

Iz razreda BMic prejema zajet zvočni zapis in podatek o glasnosti le-tega. Ob prekomerni glasnosti na pregledovalnike pošlje kontrolno sporočilo »`Event=Sound`«, prejeti zvočni zapis pa posreduje na pregledovalnike.

Ob pritisku tipke na vhodu za slušalke na pregledovalnike pošlje kontrolno sporočilo »`Event=Alarm`«.

Iz razreda BmCommunicator prejema dogodke ob prejetem zvočnem zapisu iz pregledovalnikov in le-tega predvaja na zvočnikih.

4.2.2 Razred BClient

Razred BClient (Slika 4) vsebuje objekte za upravljanje z zvočnikom (*BSpeaker*) in mikrofonom (*BMic*), objekt, ki skrbi za mrežno komunikacijo (*BcCommunicator*), ter objekt, ki skrbi za prikaz slike na aktivnosti (*BImageView*).

Iz razreda BcCommunicator prejema kontrolna sporočila, iz katerih izlušči podatke o dogodkih, zvočni zapis in sliko. V primeru, da je vklopljen monitor, prejeti zvočni zapis predvaja na zvočnikih in prejeto sliko prikaže na aktivnosti.

Iz razreda BMic prejema zajet zvočni zapis. V primeru, da je vklopljen monitor, prejeti zvočni zapis pošlje na snemalnik, ki ga spremlja na monitorju.

```
class BMaster implements BCameraEvents, BMicEvents, BmCommunicatorEvents {
    private BCamera camera;
    private BMic mic;
    private BSpeaker speaker;
    private BmCommunicator communicator;
    private BImageView view;

    // called from BCamera when the new image is available
    public void onImageCaptured(BImage img)
    {
      boolean motion = detect_motion();
      if (motion)
        communicator.sendCommand("Event=Motion");
      view.updateImage(img.getAsBitmap()); // show image
      byte[] data = img.getAsJpeg(); // compress image
      communicator.sendImage(data, 0, data.length);
    }

    // called from BMic when the new sound frame is recorded
    public void onSoundFrameRecorded(byte[] data, int pos,
                int len, double volume)
    {
      if (volume > level)
        communicator.sendCommand("Event=Sound");
      communicator.sendAudio(data, pos, len);
    }

    // called when media button is pressed
    public void onAlarmButtonPressed()
    {
      communicator.sendCommand("Event=Alarm");
    }

    // called from communicator when audio is received
    public void onBmCommunicatorAudioReceived(BmTcpClient c,
                byte[] buff, int pos, int len)
    {
      speaker.play(buff, 0, len);
    }
}
```

Slika 3: Poenostavljen razred BMaster

```
public class BClient implements BMicEvents, BcCommunicatorEvents {
    private BSpeaker speaker;
    private BMic mic;
    private BcCommunicator communicator;
    private BImageView view;

    @Override
    // called from communicator when command is received
    public void onBcCommunicatorCommandReceived(BcMaster m, String cmd) {
        // process commands
    }

    @Override
    // called from communicator when new sound frame is received
    public void onBcCommunicatorAudioReceived(BcMaster m,
                byte[] data, int pos, int len) {
      if (monitoring)
        speaker.play(data, pos, len);
    }

    @Override
    // called from communicator when new image is received
    public void onBcCommunicatorImageReceived(BcMaster m,
                byte[] data, int pos, int len) {
      if (monitoring)
        view.updateImage(BitmapFactory.decodeByteArray(data, pos, len));
    }

    @Override
    public void onSoundFrameRecorded(byte[] data, int pos,
                int len, double loudness) {
      if (monitoring)
        monitoringMaster.sendAudio(data, pos, len);
    }
}
```

Slika 4: Poenostavljen razred BClient

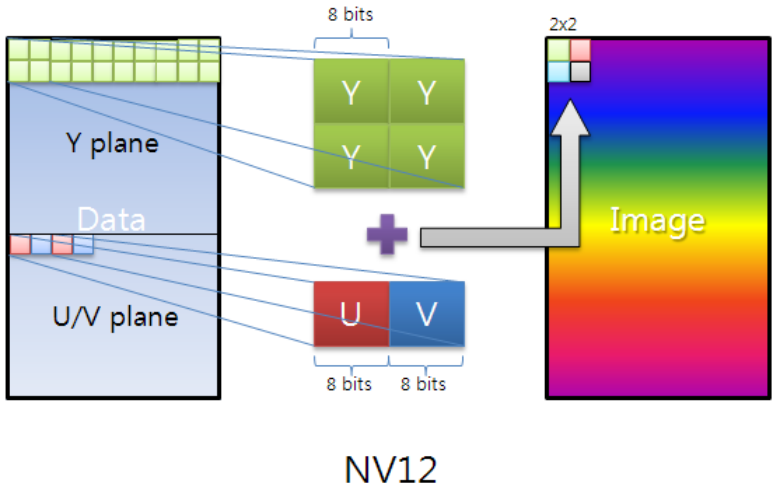
4.2.3 Zajem slike in zaznavanje gibanja

Za zajem slike smo uporabili razred `android.hardware.Camera`.

Dostop do kamere dobimo s klicem funkcije `open()`, kateri kot argument podamo zaporedno številko kamere, do katere želimo dostopati (0 – kamera na hrbtni strani, 1 – kamera na prednji strani).

Kameri smo nato nastavili parametre za zajem slike. Sliko zajemamo v velikosti 240x320 slikovnih točk v formatu NV21. Zajem slike zaženemo s klicem funkcije `startPreview()`, nato pa sistem ob vsaki zajeti sliki kliče funkcijo, ki smo jo nastavili kot funkcijo povratnega klica.

Format NV21 (Slika 5) je oblika zapisa YCbCr barvnega modela. Pri YCbCr barvnem modelu je vsaka slikovna točka predstavljena s svetlostjo (Y) ter razliko med modro barvo in svetlostjo (Cb) ter razliko med rdečo barvo in svetlostjo (Cr).



Slika 5: Format zapisa NV21[11]

Za format NV21 smo se odločili, ker je podprt od prvih verzij Androida dalje, hkrati pa že vsebuje podatke o svetlosti slikovnih točk, kar uporabimo pri zaznavanju premika.

Za zaznavanje premika uporabimo sledeči algoritem:

Sliko razdelimo na 16x16 con, nato pa v vsaki coni izračunamo povprečno svetlost. Povprečne vrednosti svetlosti con normaliziramo, nato pa odštejemo normalizirane vrednosti con trenutne slike od normaliziranih vrednosti con prejšnje slike. Absolutno vrednost razlike con nato primerjamo z mejno vrednostjo in če je absolutna vrednost razlike nad mejno vrednostjo v katerikoli coni, smo zaznali premikanje.

Preden se slika iz snemalnika pošlje pregledovalnikom, se zakodira v Jpeg format.

4.2.4 Zajem zvoka in računanje glasnosti

Za zajem zvoka smo uporabili razred `android.media.AudioRecord`.

Ob inicializaciji smo nastavili parametre za zajem enokanalnega zvoka v 16 bitnem PCM formatu z vzorčevalno frekvenco 8000 vzorcev na sekundo.

Zajem zvoka se začne s klicem `startRecording()` funkcije, podatke o zvočnem zapisu pa dobimo s klicem funkcije `read()`. Funkcija `read()` nam vrne podatkovno polje (*array*), v katerem je zapisan trenutni del zvočnega zapisa.

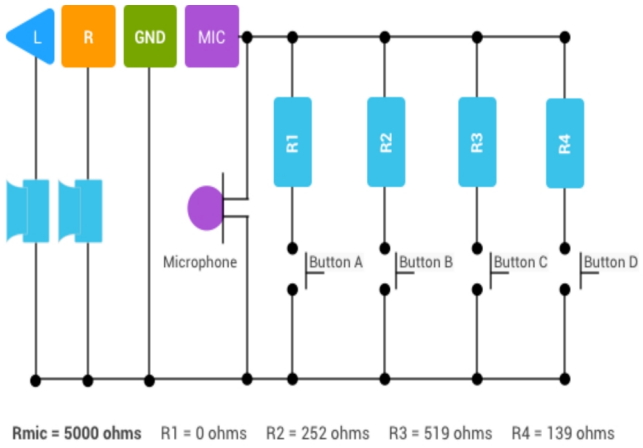
Iz podatkovnega polja izračunamo glasnost trenutnega dela zvočnega zapisa po formuli:

$$glasnost = \sqrt{\frac{1}{n}\sum_{i=0}^{n} data_i^2} \quad ,$$

kjer je n število podatkov v podatkovnem polju, $data_i$ pa vrednost i-tega elementa podatkovnega polja.

4.2.5 Detekcija pritiska tipke na vhodu za slušalke

Android pametni telefon na vhod za slušalke in mikrofon podpirajo priklop štirih tipk [12]. Slika 6 prikazuje shemo priklopa tipk na priključek za slušalke, da jih Android zazna. V naši aplikaciji podpiramo zaznavo prve tipke (*Button A*). Zaznavanje poteka z implementacijo `onReceive()` funkcije `BroadcastReceiver` razreda. Pritisk prve tipke proži dogodek *`ACTION_MEDIA_BUTTON`*. Ob tem dogodku pokličemo funkcijo `BMaster.onMediaButtonPress()`, ki na povezane pregledovalnike pošlje kontrolno sporočilo »`Event=Alarm`«.



Slika 6: Shema priklopa tipk na priključek za slušalke

4.2.6 Mrežno povezovanje

Povezovanje snemalnikov in pregledovalnikov poteka preko TCP protokola. Preden se pregledovalnik lahko poveže na snemalnik, mora le-tega najti. Iskanje snemalnikov poteka preko UDP protokola, kjer pregledovalnik vsakih 5 sekund pošlje broadcast zahtevo, na katero mu snemalniki odgovorijo. Tako si pregledovalnik gradi in osvežuje seznam aktivnih snemalnikov.

Broadcast zahteva, ki jo pošilja pregledovalnik, je niz:

»Pakiga:<skupina>«,

kjer je <skupina> ime skupine, kateri pripada pregledovalnik. Snemalnik na zahtevo odgovori le, če pripada isti skupini. Odgovori z nizom:

»Here:<ime>«,

kjer je <ime> ime snemalnika.

Ob prejemu odgovora pregledovalnik preveri seznam aktivnih snemalnikov in če snemalnika z dobljenim imenom še nima na seznamu, se, preden ga doda na seznam, nanj poveže preko TCP protokola. Po vzpostavljeni TCP povezavi pregledovalnik pošlje niz:

»Group:<skupina>\nName:<ime>\n«,

kjer je <skupina> ime skupine, kateri pripada pregledovalnik in <ime> ime pregledovalnika. Na prejeti niz snemalnik odgovori z nizom:

»Pakiga:<ime>«,

kjer je <ime> ime snemalnika. Če se ime skupine, poslano s strani pregledovalnika, ne ujema z imenom skupine snemalnika, snemalnik povezavo prekine.

4.2.7 Komunikacijski protokol

Komunikacija med snemalnikom in pregledovalnikom poteka preko podatkovnih kanalov. Podatki posameznih kanalov se prenašajo v paketih. Prvih 8 bytov paketa je glava, v kateri so podani številka pošiljatelja (CID, 1 byte), številka kanala (SID, 1 byte) in velikost paketa (N, 2 byta) v bytih. Glavi nato sledi N bytov podatkov. Številka pošiljatelja je pri neposredni povezavi med snemalnikom in pregledovalnik vedno 0.

Zgradba paketa

glava							podatki
0x3c	0x3c	CID	SID	N	0x3e	0x3e	N bytov podatkov

Obstajajo trije podatkovni kanali (SID):

- kontrolni kanal (SID = 1):

  preko kontrolnega kanala se prenašajo kontrolna sporočila. Kontrolna sporočila so nizi, ločeni z NUL znakom;

- zvočni kanal (SID = 2):

  preko zvočnega kanal se prenašajo paketi zvočnega zapisa. V prvih dveh bytih je podano število bytov posameznega paketa;

- slikovni kanal (SID = 3):

  preko slikovnega kanala se prenašajo posamezne slike. V prvih dveh bytih je podano število bytov posamezne slike.

4.2.8 Posredniški strežnik

Povezovanje snemalnikov in pregledovalnikov na posredniški strežnik poteka preko TCP protokola. Po vzpostavljeni povezavi se na strežnik pošlje niz:

»Group:<skupina>\nName:<ime>\nType:<tip>«,

kjer je <skupina> ime skupine, kateri pripada snemalnik oziroma pregledovalnik, <ime> je ime snemalnika oziroma pregledovalnika, <tip> pa je »recorder« za snemalnik oziroma »viewer« za pregledovalnik. Na prejeti niz strežnik ogovori z:

»OK«.

Strežnik nato vsem povezanim pregledovalnikom vsakih 5 sekund pošilja seznam snemalnikov, ki so v njegovi skupini. Seznam mu pošilja preko kontrolnega kanala, kjer je vrednost CID nastavljena na 0xff. Kontrolno sporočilo pa je:

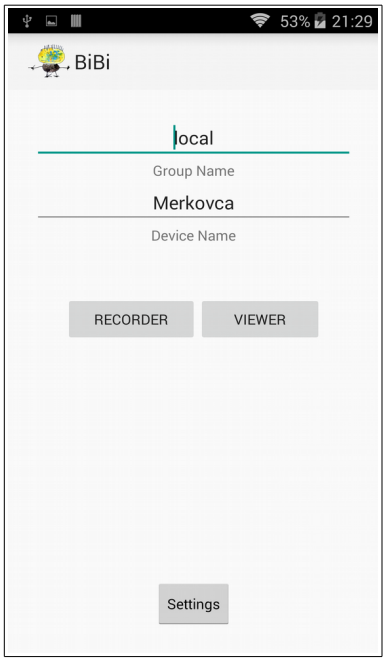
»Recorders=<ime1>\t<ime2>\t...«,

kjer je <ime1> ime prvega snemalnika, <ime2> ime drugega, itd.

Posredniški strežnik podatke, ki jih prejme od povezanih snemalnikov, posreduje na vse povezane pregledovalnike iste skupine. Pri posredovanih paketih spremeni CID na zaporedno številko snemalnika, za katerega posreduje podatke. S tem pove pregledovalniku, od katerega snemalnika izvirajo podatki.

4.3 Funkcionalna specifikacija

Ob zagonu aplikacije se odpre osnovno okno (Slika 7), kjer uporabnik določi ime naprave in skupine. S pritiskom na gumb »Settings« se odpre okno z nastavitvami. S pritiskom na gumb »Recorder« se zažene snemalnik in s pritiskom na gumb »Viewer« se zažene pregledovalnik.



Slika 7: Osnovno okno

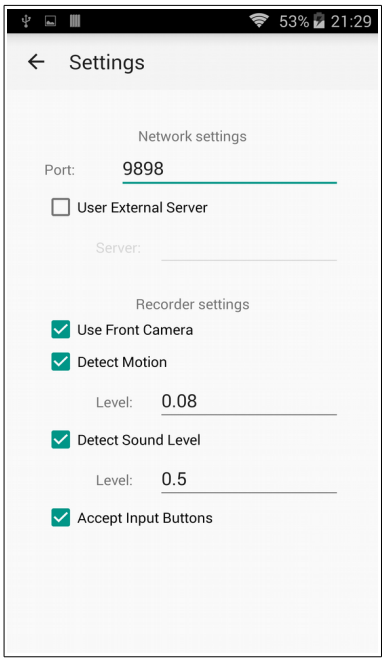
4.3.1 Nastavitve

Nastavitve (Slika 8) so razdeljene v dve skupini.

V prvi skupini uporabnik lahko nastavi omrežne nastavitve, ki veljajo za snemalnik in pregledovalnik. V omrežnih nastavitvah nastavi številko vrat (*Port*), preko katerih poteka mrežna komunikacija in omogoči uporabo posredniškega strežnika (*Use External Sever*) ter določi njegov naslov (*Address*).

Druga skupina pa vsebuje nastavitve za snemalnik. Tu lahko uporabnik izbere, ali bo snemalnik uporabljal kamero na hrbtni strani naprave, kar je privzeta vrednost, ali kamero na sprednji strani (*Use Front Camera*). Omogoči lahko funkcijo zaznavanja

premikov (*Detect Motion*), zaznavanja prekomerne glasnosti (*Detect Sound Level*) in omogoči zaznavanje pritisnjene tipke na vhodu za slušalke (*Accept Input Buttons*). Pri zaznavanju premikanja in glasnosti lahko nastavi stopnjo (*Level*) premika oziroma glasnosti, pri kateri se sproži zaznava. Vrednost je v intervalu med 0.0 in 1.0, manjša vrednost pomeni bolj občutljivo zaznavanje.

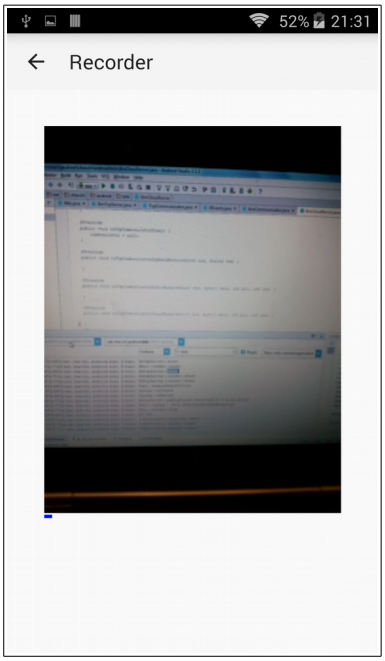


Slika 8: Nastavitve

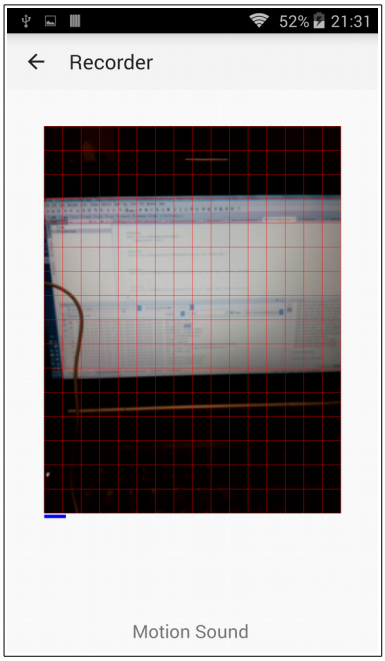
4.3.2 Snemalnik

Okno snemalnika (Slika 10) je precej preprosto in uporabniku ne ponuja nobene možnosti interakcije. Sestavljeno je iz slike, ki jo zajema kamera, pod sliko pa je modra črta, ki prikazuje trenutno jakost zvoka. V primeru, da je omogočena možnost zaznavanja premikov, je preko slike narisana mreža posameznih con (Slika 9), kjer se zaznava gibanje. V primeru zaznanega gibanja v posamezni coni, se v zgornjem levem kotu cone nariše zelen kvadratek. S tem uporabnik vidi, če je nastavljena stopnja zaznavanja premika zadostna.

V primeru katerega od zaznanih dogodkov, se le-ti izpišejo na spodnjem delu okna.



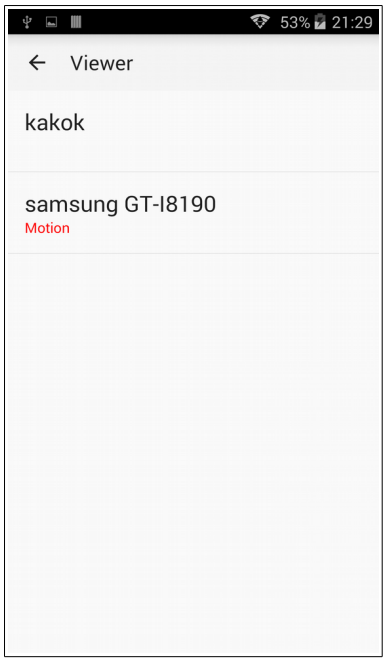
Slika 10: Snemalnik



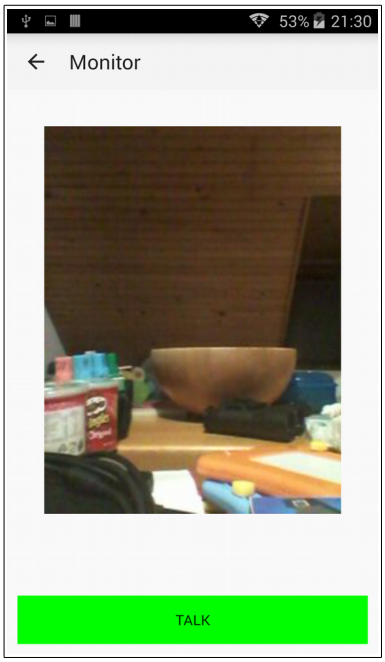
Slika 9: Snemalnik z zaznavanjem premikanja

4.3.3 Pregledovalnik

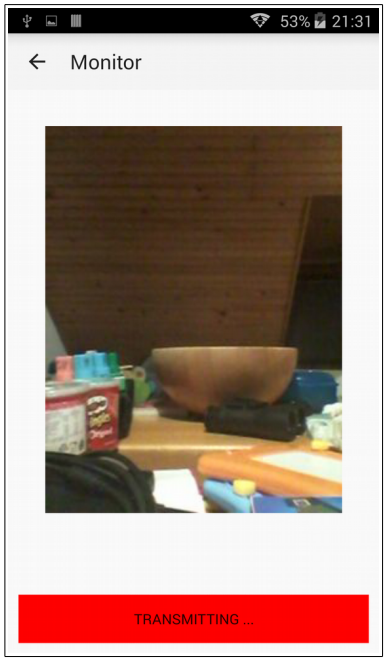
Pregledovalnik vsebuje seznam povezanih snemalnikov (Slika 12). Za vsak snemalnik je prikazano njegovo ime in pod imenom morebitni dogodki, ki jih je snemalnik zaznal. Uporabniku omogoča izbiro posameznega snemalnika, s čimer se odpre okno monitorja (Slika 12). Monitor prikazuje sliko in predvaja zvok, zajet na snemalniku. Na spodnji strani okna se nahaja gumb »Talk«, s katerim omogočimo dvosmerno komunikacijo s snemalnikom (Slika 11).



Slika 13: Pregledovalnik



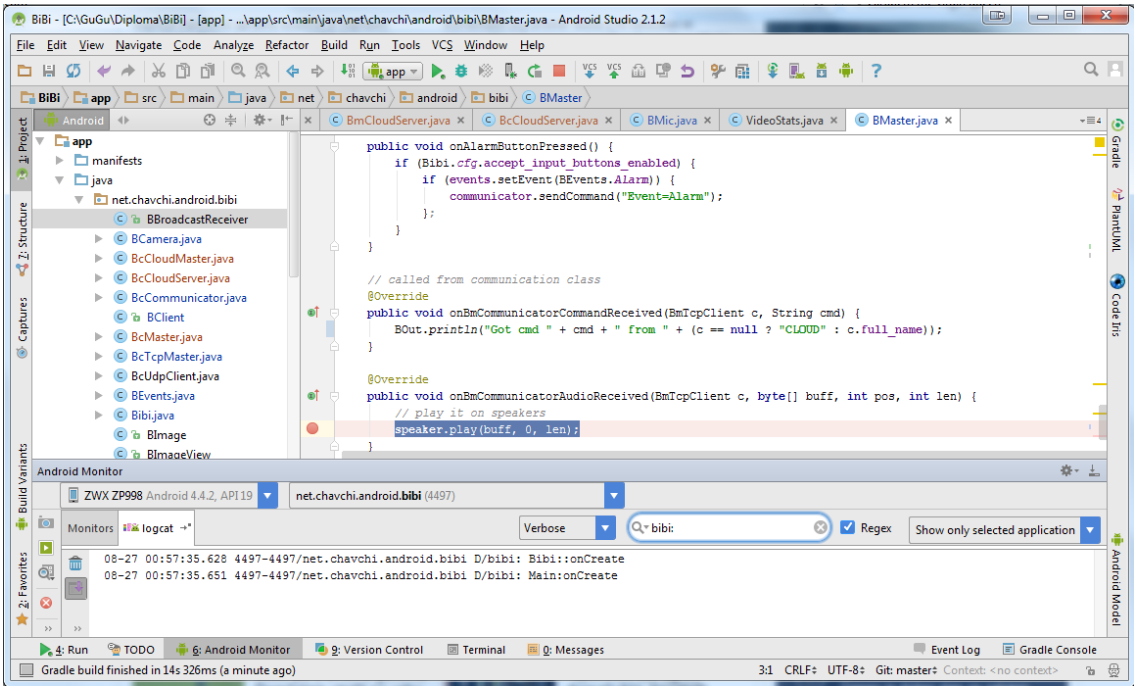
Slika 12: Monitor



Slika 11: Monitor z omogočeno dvosmerno komunikacijo

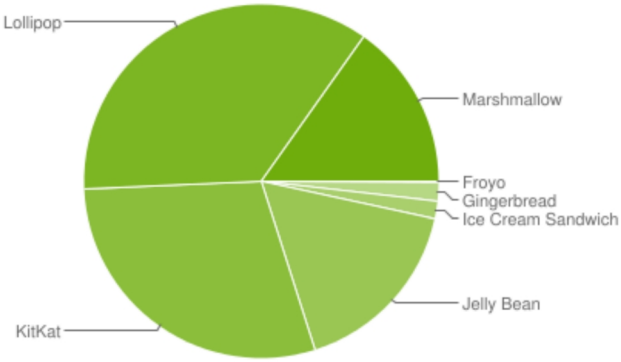
5. Implementacija

Za sam razvoj smo uporabili razvojno okolje Android Studio verzije 2.1.2 (Slika 14). Razvojno okolje nam ponudi čarovnika, preko katerega nastavimo osnovne atribute projekta, kot so: ime aplikacije, ime paketa in pot do lokacije na disku, kjer bo projekt shranjen. Potrebno je določiti tip aplikacije in minimalno podprto API verzijo.



Slika 14: Android studio

Zaradi želje, da bi aplikacija tekla na kar največjem možnem številu pametnih telefonov, smo za minimalno API verzijo uporabili verzijo 8, torej Android 2.2 (Froyo). Android 2.2 je bil izdan leta 2010 in je bil ob času pisanja naše aplikacije star 6 let. Poganjal je le še 0,1% Android naprav. Vse ostale Android naprave je poganjala ena od novejših verzij. To pomeni, da naj bi naša aplikacija tekla na (skoraj) vsakem Android telefonu, ki je danes v uporabi (Slika 15).



Slika 15: Tržni delež Android verzij [9]

Same izdelave aplikacije smo se lotili s spoznavanjem razvojnega orodja in potekom razvoja Android aplikacij. Nekaj časa smo namenili spoznavanju življenjskega cikla posameznih aktivnosti v aplikaciji ter pravilnemu povezovanju samih elementov uporabniškega vmesnika. Naslednja stvar je bila dostop do mikrofona, zvočnika ter kamere. Android razvojno okolje ima zelo dobro dokumentacijo [15], kjer se poleg opisov knjižnic in funkcij nahajajo tudi podrobna navodila za dostop in uporabo posameznih naprav znotraj telefona. S pomočjo te dokumentacije je bilo sorazmerno lahko implementirati zajem ter predvajanje zvoka. Pri zajemu in prenosu zvoka smo želeli uporabiti kodirnik, ki bi nam zmanjšal količino podatkov, vendar pa API verzija 8 te funkcionalnosti še ne podpira, tako da smo ostali pri nestisnjenem zvočnem zapisu.

Z API verzijo smo bili omejeni tudi pri delu s kamero. Andorid je v API verziji 21 dodal nov vmesnik za delo s kamero *camera2,* ki ponuja nekoliko več svobode in možnosti za delo s kamero. Mi smo uporabili (od API verzije 21 naprej opuščen) vmesnik *Camera*. Odločili smo se za velikost slike 240x320 slikovnih točk ter format zapisa NV21. Za format smo se odločili, ker je to eden izmed formatov, podprtih v starejših verzijah Androida, in ima informacijo o svetlosti zapisano v ločenem kanalu. To smo uporabili pri razvoju algoritma za detekcijo gibanja, saj sam algoritem temelji na spremembi svetlosti v posameznih delih slike.

Po uspešni implementaciji zajema in predvajanja zvoka ter zajema slike smo se lotili mrežne komunikacije. Za samo delo z mrežnimi vtičniki ima programski jezik Java zelo dobro podporo, tako da nam je večino časa razvoja komunikacije vzela implementacija

protokola za prenos podatkov. Pri testiranju mrežne komunikacije nam je bilo v veliko pomoč dejstvo, da smo obe funkcionalnosti (snemalnik in pregledovalnik) vključili v eno aplikacijo, saj smo tako lahko z eno instanco Android Studia, na katerega smo imeli povezane tri telefone, poganjali in razhroščevali tako snemalni kot pregledovalni del aplikacije.

Kot dodatek smo se lotili razvoja posredniškega strežnika, ki omogoča, da s telefonom pregledujemo snemalnike, tudi ko nismo priključeni v lokalno omrežje. Strežnik smo razvili kot samostojen javanski program. Pri razvoju strežnika smo morali precej spremeniti v sami aplikaciji, saj smo morali prikazovalniku dodati možnost prikazovanja snemalnikov, ki se ne nahajajo v lokalnem omrežju, kar pa v osnovni zgradbi aplikacije ni bilo predvideno.

Izvorno kodo aplikacije smo objavili na internetu in je dosegljiva na naslovu https://github.com/bibi-2016/bibi

5.1 Praktična uporaba

Med razvojem aplikacije smo za potrebe testiranja aplikacijo uporabili kot elektronsko varuško. Mobitel na katerem je aplikacija tekla v načinu snemalnika smo postavili v otroško sobo, na drugem telefonu pa smo aplikacijo pognali v načinu monitorja, ter tako spremljali popoldanski spanec otrok. Ponudila se nam je celo priložnost, ko smo v isti hiši imeli speče otroke dveh družin in smo aplikacijo poganjali na štirih telefonih, pri čemer smo imeli dva telefona v eni skupini in druga dva v drugi, tako da je vsak starš lahko spremljal samo svojega otroka. Aplikacija je svoje delo opravila po pričakovanjih, saj smo lahko slišali kdaj so se otroci prebudili.

6. Zaključek

Cilj diplomskega dela je bil razviti aplikacijo, namenjeno odsluženim pametim telefonom, ki bi jo lahko uporabili za namen nadzora hiše. S končnim rezultatom smo precej zadovoljni, saj aplikacija omogoča povezavo med več kot enim snemalnikom in pregledovalnikom v lokalnem brezžičnem omrežju. Odkrivanje snemalnikov in povezovanje na njih znotraj lokalnega omrežja deluje tekoče, pri prenosu slike in zvoka pa občasno prihaja do krajših motenj. Poleg medsebojnega povezovanja znotraj lokalnega omrežja aplikacija omogoča tudi povezavo na posredniški strežnik zunaj lokalnega omrežja ter s tem omogoča oddaljeni nadzor nad telefoni znotraj hiše.

Trenutna aplikacija za samo mrežno komunikacijo ne uporablja nobene zaščite pred prestrezanjem prometa, tako da bi bilo za resno uporabo potrebno komunikacijski protokol nadgraditi s SSL/TLS enkripcijo, ki bi omogočala večjo zasebnost pri prenosu podatkov. Prav tako aplikacija ne omogoča privatnih skupin, kjer bi bilo pri povezavi na snemalnik potrebno vnesti geslo. To bi dodalo nivo uporabnosti, saj bi v istem omrežju lahko poganjalo aplikacijo več neodvisnih skupin. Pri sami aplikaciji bi bilo potrebno nagraditi opozorila na dogodke z zvočnim signalom ter spremeniti aplikacijo tako, da bi omogočala prejemanje dogodkov, tudi ko ni aktivna na telefonu. Aplikacijo dejansko lahko uporabljamo za preprost nadzor znotraj hiše in v dobrih svetlobnih pogojih, poskrbeti moramo le, da telefon priključimo na napajanje, saj je sama aplikacija energijsko precej potratna. Za zahtevnejši nadzor pa pametnih telefonov ne moremo uporabiti, saj jih zaradi pomanjkanja ustrezne zaščite ne moremo uporabiti zunaj hiše, v slabših svetlobnih pogojih pa nam za nadzor ostane le zvok.

Literatura

[1] (2016) The "Only" Coke Machine on the Internet. Dostopno na: https://www.cs.cmu.edu/~coke/history_long.txt

[2] (2016) That »Internet of Things« thing. Dostopno na: http://www.rfidjournal.com/articles/view?4986

[3] (2016) Internet of Things Installed Base. Dostopno na: http://www.gartner.com/newsroom/id/2636073

[4] T. T. Mulani, S. V. Pingle, Internet of Things, IRJMS & SPPP's, marec 2016, zv. 2

[5] H. Kato, K.T. Tan, Pervasive 2D Barcodes for Camera Phone Applications, IEEE Pervasive Computing, oktober 2007, zv. 6 str. 76-85

[6] (2016) Kaj je IoT. Dostopno na: http://www.stroka.si/Novice/poglabljamo-nac5a1e-znanje-o-internet-of-things-iot-2331

[7] R. Want, B. N. Schilit, S. Jenson, Enabling the Internet of Things, IEEE Computer Society, januar 2015, zv. 48, str. 28-35

[8] N. Gershenfeld, R. Krikorian, D. Cohen, The Internet of Things, Scientific American, oktober 2004, str.. 76-81

[9] (2016) Android Platform Versions. Dostopno na: https://developer.android.com/about/dashboards/index.htm

[10] (2016) Understanding the Activity lifecycle. Dostopno na: http://www.xda-developers.com/everything-devs-need-to-know-about-multiwindow-in-android-n-code-examples/

[11] (2016) NV21 Explanation. Dostopno na: http://stackoverflow.com/questions/22456884/how-to-render-androids-yuv-nv21-camera-image-on-the-background-in-libgdx-with-o

[12] (2016) Wired Audio Headset Specification. Dostopno na: https://source.android.com/devices/accessories/headset/specification.html

[13] (2016) Android pre-histroy. Dostopno na: http://www.androidcentral.com/android-pre-history

[14] (2016) Smartphne market share by operating systems. Dostopno na: http://www.statista.com/statistics/272307/market-share-forecast-for-smartphone-operating-systems/

[15] (2016) Introdution to Android. Dostopno na: https://developer.android.com/guide/index.html